

REMARKS

This Amendment is requested to be filed in connection with an RCE filed in response to the Final Office Action dated January 6, 2005. In the Office Action, the Examiner rejected claims 1-6, 8-13, 15-25, and 27-29 under 35 U.S.C. § 102(b) as being anticipated by Ishii et al., U.S. Patent No. 5,835,761 (hereinafter *Ishii*).

Claims 1, 4-7, 9, 11, 14-18, 20, 21, 23, and 26-29 are amended as shown above. Specifically, independent claims 1, 9, 18, and 21 are amended to more clearly recite features of the claimed invention. Claims 1-29 remain pending in the application. For the reasons set forth below, the Applicants respectfully request reconsideration and allowance of all pending claims.

It is noted that many of the claims have been amended to replace the term “original” with existing. The motivation for this amendment was that the term “original” could be interpreted (wrongly) as being limited to only that portion of firmware that was originally provided with a computer system. The term “existing” implies that the portion of firmware to be replaced comprises firmware that currently exists – this firmware may comprise firmware that was originally provided with a computer system or firmware that has been previously updated.

Argue in Support of Allowance of claims - 35 U.S.C. § 102

A claim is anticipated only if each and every element of the claim is found in a single reference. M.P.E.P. § 2131 (citing *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628 (Fed. Cir. 1987)). “The identical invention must be shown in as complete detail as is contained in the claim.” M.P.E.P. § 2131 (citing *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226 (Fed. Cir. 1989)).

In support of the rejection of claim 1, in its original form, the Examiner asserts *Ishii* discloses:

writing updated firmware data that is to replace the original portion of platform firmware data to a firmware storage device (col. 3 lines 22-27);

and atomically modifying firmware configuration data to indicate the updated firmware data is to be used in place of the original portion of platform firmware data that is being updated such that only all of the original portion of platform firmware data are valid prior to the atomic modification of the firmware configuration data and only all of the updated firmware data are valid after the atomic modification to the firmware configuration data (col. 3 lines 22-55, col. 9 lines 30-67, and col. 10 lines 24-36).

Claim 1 has been amended to recite, in pertinent part,

writing updated firmware data that is to replace the existing portion of platform firmware data to a firmware storage device in which the existing portion of firmware data are stored; and

atomically modifying firmware configuration data in the firmware storage device to indicate the updated firmware data is to be used in place of the existing portion of platform firmware data that is being updated such that only the existing portion of platform firmware data are valid prior to the atomic modification of the firmware configuration data and only the updated firmware data are valid after the atomic modification to the firmware configuration data.

Ishii discloses an information processing system capable of updating a BIOS programme without interrupting or stopping the operation of the system. Various embodiments for accomplishing this task are disclosed. However, each embodiment generally follows the same sequence. The updated BIOS programme is first written to a temporary storage area, which in the embodiment of Figure 5, for example is update program memory 48. Meanwhile, the original BIOS programme is stored in non-volatile BIOS memory 45. During initialization operations, the original BIOS programme in BIOS memory 45 is written to shadow RAM in main memory 44, as was a common practice at the time the *Ishii* patent application was filed. After a multi-tasking operating system (OS) is loaded, the OS is used to enable an updated BIOS program stored on an external medium (e.g., flexible disc 53), device, or network) to be copied to a temporary store (e.g., update program memory 48 in Figure 5 or first common

memory 67a in the embodiment of Figure 15) on information processing system 41. The original BIOS program (stored in BIOS memory 45) is then subsequently replaced with the updated BIOS program under management of the OS.

Each embodiment disclosed by *Ishii* employs a pair of firmware storage devices and a separate memory switch unit 47. For instance, each of the embodiments of FIGs. 5, 11-14, 23, and 29-33 employ a recovery memory 46 and a BIOS memory 45, in addition to a separate update programme memory 48 and a memory switch unit 47. Each of the embodiments of FIGs. 15, and 19-22 employ a first common memory 67a and a second common memory 67b in combination with the memory switch unit 47.

In consideration of the current amendment to claim 1, it is clear that *Ishii* does not teach the element of “atomically modifying firmware configuration data in the firmware storage device to indicate the updated firmware data is to be used in place of the existing portion of platform firmware data that is being updated such that only the existing portion of platform firmware data are valid prior to the atomic modification of the firmware configuration data and only the updated firmware data are valid after the atomic modification to the firmware configuration data.”

Ishii does not modify firmware configuration data in a firmware storage device to perform a similar operation. Rather, in the embodiments that employ update programme memory 48, the only configuration data that is changed is the BIOS update flag 50. In the embodiments that employ the common memories, the only configuration data that are changed pertain to programmes (either BIOS or recovery programmes) stored in an alternate common memory, as discussed below in further detail.

When a computer system of the type used by *Ishii* is booted (initially started or reset), the processor looks to a specific address from which the BIOS is loaded. During an initialization operation, the BIOS code is shadowed into much faster system RAM (see, e.g., col. 10, lines 23-38).

As stated in col. 9, lines 15-30,

The BIOS memory 45 is for storing the basic input/output system programmes while the recovery memory 46 is for storing programmes for use in performing operations required to restore the system when a failure occurs. The BIOS memory 45 and the recovery memory 46 are non-volatile memories of which contents may be rewritten on the board. These memories may be formed of, for example, flash memories. As described above, flash memories are derived from EEPROMs of which contents may be rewritten on the board of the information processing system. The memory switch unit 47 is for use in switching between the BIOS memory 45 and the recovery memory 46. More specifically, the memory switch unit 47 switches the BIOS memory 45 into the recovery memory 46 when the information processing system 41 fails to load the basic input/output system programmes stored in the BIOS memory 45. (Emphasis added)

Ishii clearly does not perform an atomic update in the manner that anticipates claim 1. Details of *Ishii*'s update scheme are shown in the flowcharts of FIGs. 6-10. During the operations of FIG. 6, an entire BIOS programme comprising the updated BIOS is loaded into update programme memory 48 (step A4), and the BIOS update flag 50 is set (step A5). (The BIOS update flag is in the update programme memory 48 – e.g., see FIG 5.) Similar operations are shown in FIG. 7 for loading a recovery programme update into the update programme memory 48. In FIG. 8, the computer system is reset and initiated. At step C5, the data in BIOS memory 45 is copied in main memory 44 as a shadow copy of the BIOS. Henceforth, the computer system uses the copy of the BIOS in memory 45. If updating of the BIOS is indicated by setting of the update flag 50, the process proceeds to FIG. 9. At step D1, the updated BIOS programme that was previously written into the update programme memory 48 is written into BIOS memory 45. This comprises overwriting the existing BIOS memory in the BIOS memory 45. Since this memory is a flash memory, the only way to write the data is to clear or set (depending on the flash type) all of the bits in all of the blocks that are to be employed for storing the new BIOS programme, and this to toggle selected bits in those blocks. It is noted that flash memory, at least at the time of the *Ishii* invention, does not allow an individual bit to be toggled more than once without having to “erase” an entire block. (See present application, page 8, last paragraph).

Thus, this requires the existing BIOS programme to be first erased and then subsequently replaced by writing new data to the erased blocks. Thus, there is no way that *Ishii* can teach the operation of “atomically modifying firmware configuration data in the firmware storage device to indicate the updated firmware data is to be used in place of the existing portion of platform firmware data that is being updated such that only the existing portion of platform firmware data are valid prior to the atomic modification of the firmware configuration data and only the updated firmware data are valid after the atomic modification to the firmware configuration data.” When the updated BIOS is written in the firmware storage device (e.g., BIOS memory 45), the prior existing version of the BIOS programme that is being replaced no longer exists.

This could be very problematic if a failure occurred during the update, or if somehow the data corresponding to the new BIOS programme was corrupt. *Ishii* addresses this problem by employing a separate storage device for storing another BIOS programme in recovery memory 46. This is also referred to as the recovery programme. The recovery programme is employed in case the BIOS programme is corrupted or otherwise inaccessible, as follows.

In response to a system reset in FIG. 8, a normal activation of BIOS memory 45 is performed, followed by a checksum of the BIOS programme. If the normal activation fails (at step C4) or checksum fails (at step C4 in FIG. 8), the recovery memory 46 is activated by memory switch 47. The recovery programme stored in the recovery memory is used to recover the system in the event that the BIOS programme in BIOS memory 45 fails.

Thus, it is clear that there is no circumstance under *Ishii* for which an atomic modification is made to firmware configuration data such that existing platform firmware data are valid prior to the atomic modification and updated firmware data are valid after the modification. The only time that both an existing BIOS programme and an updated BIOS program exist at the same time are when the existing BIOS programme is present

in BIOS memory 45 and the updated BIOS programme is present in update program memory 48. Any BIOS programme in update programme memory 48 is not valid because it cannot be loaded – the BIOS programme must be in BIOS memory 45 to be loaded (and thus be valid). Accordingly, there is no atomic operation that invalidates an existing firmware data while at the same time validating updated firmware data.

With respect to the embodiments employing a first common memory 67a and a second common memory 67b, in these embodiments the updated BIOS programme and recovery programme are written to these common memories rather than an update programme memory. However, there is no circumstance under which a BIOS programme and its replacement (updated BIOS programme) are written to the same common memory 67a or 67b. Rather, they are stored in an alternating manner. As stated in col. 16, lines 3-24,

FIG. 15 shows an information processing system according to a sixth embodiment of the present invention. This information processing system is similar to that described in conjunction with the first embodiment except that a common memory is provided for storing both the basic input/output system programmes and the recovery programmes in place of the update programme memory 48 in FIG. 5. As the common memory, first and second common memories are provided to ensure alternate storing of new programmes. As shown in FIG. 15, the information processing system 41 comprises first and second common memories 67a and 67b, first and second storage areas 68a and 68b, and first and second valid flags 69a and 69b. The first and the second storage areas 68a and 68b are part of the first and the second common memories 67a and 67b, respectively. Likewise, the first and the second valid flags 69a and 69b are part of first and second common memories 67a and 67b, respectively, and are indicative of the common memory in which the newer basic input/output system or recovery programme is stored. The first and the second common memories 67a and 67b are non-volatile memories such as flash memories, of which contents may be rewritten on the board. (Emphasis added)

Further details of how this embodiment operates are shown in the flowcharts of FIGs. 16-18. In general, a valid flag in one of the first and second common memories is used to identify which version (the existing or updated BIOS programme) is valid at any point in time. However, as discussed above, the existing and updated BIOS

programmes are not stored in the same common memory, but are stored in separate common memories. This clearly does not read on the element of, writing updated firmware data that is to replace the existing portion of platform firmware data to a firmware storage device in which the existing portion of firmware data are stored.

With further respect to claim 1, this claim pertains to a “method for updating an existing portion of platform firmware data.” That is, the method enables part of an existing set of platform firmware data (e.g., part, but not all of the BIOS) to be updated in an atomic manner. It is clear that *Isiii* updates the entire BIOS programme.

As discussed in the Background of the Invention section of the present application, which discusses the prior art,

... Typically, the BIOS code will be stored as a *monolithic chunk* of code that gets replaced in its entirety by a new monolithic chunk of code. When the BIOS code is stored on a flash component, the memory blocks corresponding to the portions of memory that are to contain the new BIOS code must first be cleared (*i.e.*, reset to all 1's) prior to rewriting the memory. This clearing process wipes out the existing BIOS code. As a result, if a failure occurs in the middle of a rewrite or update, the BIOS code will be corrupt. (Last paragraph on page 3, emphasis added)

Updating an entire BIOS program is a moderately uncomplicated task. However, embodiments of the present invention claimed herein are used to atomically update one or more portions of the platform firmware. This is a much more complicated task.

More specifically, the portions of platform firmware may be stored in a single device, such as flash memory, or across multiple devices, using an Extensible Firmware Interface (EFI) that was developed by the assignee of the present application (Intel Corporation) and is now an industry standard. Details of EFI and associated firmware storage schemes are generally discussed under the Extensible Firmware Interface and Firmware Volumes section (page 7) of the present application, as follows:

Recently, the Intel Corporation has introduced a new firmware paradigm that enables firmware storage to be extended beyond the traditional monolithic storage schemes found in the prior art. This is enabled, in part, by the Extensible Firmware Interface, or EFI. As its

name indicates, the EFI enables firmware to be “extended” through use of a standardized software interface.

One means for extending firmware is facilitated by a standard software abstraction for a firmware storage device, known as a Firmware Volume (FV). Since the FV firmware storage abstraction is not tied to any specific type of hardware, it can be used to produce firmware components to the BIOS from almost any type of firmware device. For example, in a given system, one FV may represent a Flash Memory part, while another may represent a disk partition, while yet a third may represent a remote directory on a server. A single computer system may have one or more FV’s stored on one or more types of hardware.

The portions of the BIOS firmware code that are part of a FV are managed by a Firmware File System (FFS). The FFS enables one to manipulate firmware files that make up a FV. The FFS may be used for retrieving, creating, updating, and deleting firmware files. Generally, a FFS may be stored on any persistent memory device, including flash devices, disk partitions, and remote storage devices accessed via a network.

The method invention of independent claim 1 enables a portion of the platform firmware (e.g., one or more firmware files in a firmware volume) to be replaced in an atomic manner, wherein an original portion of firmware data is used prior to the atomic modification, and the updated portion of firmware data is used after the modification. This is clearly not taught or suggested by *Ishii*. At the time of filing the *Ishii* patent, EFI had yet to be invented (in fact, EFI wasn’t developed until several years later). Furthermore, there was no scheme for supporting an extensible firmware environment prior to EFI, to the best of the inventors’ knowledge. Clearly, such an environment is not disclosed by *Ishii*.

In summary, it is clear the *Ishii* does not teach each and every element of recited in independent amended claim 1, as required by a § 102 anticipation rejection. Accordingly, a rejection of amended claim 1 as anticipated by *Ishii* would be improper. Furthermore, applicants respectfully assert that a § 103 obvious rejection in view of *Ishii* alone, or in combination with and any other prior art cited in the Office Action would also

not be supported for similar reasons presented above in support of the allowance of claim 1.

In the prior office action response (mailed September 22, 2004), the Applicants asserted that *Ishii* updates an entire BIOS program and not only a portion of platform firmware. In response the Examiner states that "it is inherent that the BIOS programs could be either the entire program or just part of it (a BIOS program) on whether it needs to be updated or not." In this context, Applicant's position is that under the claimed invention of claim 1, a portion of existing BIOS may be replaced, while another portion or existing portions do not have to be replaced. Under *Ishii*, there is an all or nothing approach. Either an entire BIOS programme is updated, or an entire existing BIOS programme is employed. There is no mechanism disclosed in *Ishii* for updating a portion of platform firmware. This is particularly true since modular platform firmware did not exist at the time of the *Ishii* invention, but rather the BIOS code comprises a single program that was not modular. Furthermore, as discussed above, the entire BIOS programme being replaced is overwritten by the entire updated BIOS programme.

With respect to the allowability of amended independent claim 9, this claim now recites:

9. A method for updating a plurality of existing platform firmware files, comprising:

creating a temporary file in a firmware storage device in which the existing platform firmware files are stored;

writing data corresponding to a plurality of updated platform firmware files comprising new versions of the plurality of existing platform firmware files to the temporary file; and

modifying platform firmware file configuration information in the platform storage device to indicate that the updated platform firmware files are to be used in place of the existing platform firmware files such that only the existing platform firmware files or only

the updated platform firmware files are valid at any point in time during an update process. (Emphasis added)

It is clear from the foregoing argument in support of the patentability of amended claim 1 that *Ishii* does not store both existing platform firmware files and updated platform firmware files in the same firmware storage device. Rather, each of the *Ishii* embodiments, the existing BIOS programme and updated BIOS programme are stored in separate storage devices (when both versions of the BIOS programme are actually stored on the computer system at the same time).

Furthermore, *Ishii* clearly does not disclose updating multiple platform firmware files. Rather, *Ishii* discloses updating a single file (the BIOS programme). *Ishii* also does not employ any platform firmware file configuration information for multiple platform firmware files. The only configuration information *Ishii* employs are flags corresponding to the state of the update of the entire BIOS programme. Accordingly, the rejection of amended independent claim 9 as anticipated by *Ishii* would be improper.

With further respect to claim 9, the Examiner states in response the Applicants' assertion in an earlier office response that *Ishii* does not employ multiple platform firmware files,

In response to applicant's arguments that the references fail to show certain features of the applicant's invention, it is noted that the features upon which applicant relies (i.e., "using an Extensible Firmware Interface (EFI)") are not recited in the rejected claims(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

The Applicants agree with the statement "although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims." Thus, there is no requirement to have EFI recited in the claims. The position of the Applicants is that claim 9 requires the use of multiple platform firmware files, and such a scheme was not and could not have been employed by *Ishii*. Furthermore, the use of the EFI framework is disclosed as a preferred embodiment, but does not limit the scope

of claim 9 to only read on platform firmware that employs the EFI framework. Rather, the language of claim 9 should be interpreted by its clear meaning – that is, that a plurality of platform firmware files are being updated by, in part, writing those new files to the same firmware storage device that is being employed to store corresponding firmware files that are to be updated. This is clearly not taught by *Ishii*.

Independent claim 18 is a Beauregard claim (machine-readable medium) claiming software/firmware for performing the method of amended claim 1. Accordingly, claim 18 is patentable over the cited art for similar reasons presented above in support of allowance of independent claim 1. Similarly, Independent claim 21 is a Beauregard claim claiming software/firmware for performing the method of amended claim 9. Accordingly, claim 21 is patentable over the cited art for similar reasons presented above in support of allowance of independent claim 9.

Overall, none of the references singly or in any motivated combination disclose, teach, or suggest what is recited in the independent claims. Thus, given the above amendments and accompanying remarks, independent claims 1, 9, 18, and 21 are now in condition for allowance. The dependent claims that depend directly or indirectly on these independent claims are likewise allowable based on at least the same reasons and based on the recitations contained in each dependent claim.

If the undersigned attorney has overlooked a teaching in any of the cited references that is relevant to the allowability of the claims, the Examiner is requested to specifically point out where such teaching may be found. Further, if there are any informalities or questions that can be addressed via telephone, the Examiner is encouraged to contact the undersigned attorney at (206) 292-8600.

Charge Deposit Account

Please charge our Deposit Account No. 02-2666 for any additional fee(s) that may be due in this matter, and please credit the same deposit account for any overpayment.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN

Date: April 6, 2005

R. Alan Burnett
R. Alan Burnett
Reg. No. 46,149

12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025-1030